XML Conceptual Review

Modules Roadmap: You Are Here

- Anatomy of an XML Exchange
- XML Conceptual Review
- Basic XML Schema for NIEM
- Advanced XML Schema for NIEM
- Substitution Groups
- Extension Schemas

Objectives Roadmap

This module supports the following course objectives:

Define the physical components of an XML exchange.

Identify basic XML components that are used in the NIEM structure.

Write and/or extend an XML schema conformant to the NIEM Naming and Design Rules (NDR).

Module Objectives

- In this module, we will review how to:
  - Compare and contrast XML elements and attributes.
  - Describe the characteristics of a well-formed instance.
  - Create an XML instance based on the given XML schema.

What do XML tags (elements) do?

- The information in an XML document is contained within the various tags that constitute the markup of the document.
  - Makes it easy to distinguish the information from the markup.
  - Make it possible for incompatible computers, operating systems, & applications to share information.
  - Allow XML to become a universal translator between systems.

Naming Rules For XML Elements

- Can contain letters, numbers, and other characters.
- Must not start with number or punctuation.
- Must not start with **xml**, **XML**, or **Xml**.
- Cannot contain spaces.

Valid/Invalid Element Names

Valid

- <Person_Name>
- <Person123>
- <Person-Name>
- <Person.Name>

Invalid

- <Person_Name&>
- <123Person_Name>
- <Person Name>
- < Person_Name>
- <Person;Name>

Other Element Naming Considerations

- Make names descriptive.
- Avoid [–] and [.] and [:] in names (dash, period, and colon).
  - Usually reserved for namespaces.
  - Allowed, but not a general best practice.
- Can be as long as you like.
  - Should be descriptive and be consistently used.
- Non-English letters like “é ò á” are legal.
  - Watch out for problems… may not be supported.

Root, Parent, and Child Elements

- First element in an XML document is called a root element
- Person is the name of the element
- Also the Parent element of "PersonName"

- Parent element of "PersonGivenName" and "PersonSurName"
- Child element of "Person"

```
<Person>
  <PersonName>
    <PersonGivenName>
      Marge
    </PersonGivenName>
    <PersonSurName>
      Simpson
    </PersonSurName>
  </PersonName>
</Person>
```

XML Comments

- Examples

```
<Person>
    <!-- This is a valid comment -->
    <PersonName>
        <PersonSurName>Simpson</PersonSurName>
        <PersonGivenName>Homer</PersonGivenName>
    </PersonName>
    <!-- This is also
         a valid comment -->
    <Employment>
        <EmployerName>Springfield Nuclear</EmployerName>
    </Employment>
</Person>
```

Processing Instructions

- Generally speaking, used to encode application-specific data.
  - Most popular usage to associate presentation or transformation files with the data.
- Starts with "**<?**"
- Ends with "**?>**"
- Examples:

```
<?xml-stylesheet href="mystyle.css" type="text/css"?>

<?perl lower-to-upper-case?>
```

Prolog

- A special processing instruction included at the beginning of an XML instance.
  - Specifies version & character encoding
- Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Person>
    <PersonName>
        <PersonSurName>Flanders</PersonSurName>
        <PersonGivenName>Ned</PersonGivenName>
    </PersonName>
</Person>
```

XML Attributes

- Part of an element that provides additional information about that element.
  - Defined as a name/value pair (Single Property for an Element).

```
<?xml version="1.0" encoding="UTF-8"?>
<Person>
    <PersonName nameCode="RealName">
        <PersonSurName>Flanders</PersonSurName>
        <PersonGivenName>Ned</PersonGivenName>
    </PersonName>
</Person>
```

Quotation Marks in Attributes

- Values must always be enclosed in quotes.
  - Either ‘single’ or “double” quotes can be used.
  - Double quotes most common
  - If value contains single quotes, it is necessary to use double quotes.

Attributes Quoting Examples

```
<Person>
   <PersonName>
     <PersonSurName>Simpson</PersonSurName>
     <PersonGivenName>Homer</PersonGivenName>
   </PersonName>
   <SourceList name="Moe's Tavern Bowling Team"/>
</Person>
```

Attribute Usage

- No concrete rule to differentiate attribute usage from element usage.
- Ultimately, it is a subjective decision
  - If it seems like data, it is an element.
  - If it seems like a description of data, it is an attribute.
- **The deciding factor might be that you do not put "content" in an attribute.**

Element vs. Attribute

- An element can be any data type
- An attribute is just a string.
- An element can have other elements or attributes nested inside of it.
- An attribute ***cannot*** have an element or attribute nested inside of it.

Exercise 2.1: Attribute or Element?

- Are the following items **Attributes** or **Elements**, and why?
  - Name
  - Measurement Units
  - Age
  - Reliability
  - Accuracy

Solution 2.1: Attribute or Element?

- Name (Element)
  - How name is represented can vary significantly. It can be a simple name or a structured name.
- Measurement Units (Attribute)
  - Generally it is a quantifier. E.g., speed would use an attribute such as this to designate MPH (Miles per Hour, Foot-Pounds, Newton Meters, etc.)
- Age (Element)
  - Besides potentially needing to be extended to represent a range instead of a discrete age, qualifying the age would be easily done with something like a measurements attribute from above. E.g., Years, Days, Months.
- Reliability (Attribute)
  - It will typically qualify something else, like a statement of fact.
- Accuracy (Attribute)
  - It will typically qualify something else, like a measurement of something. E.g., 20 MPH +/- 5.

Well-formed XML *(1 of 2)*

- An XML instance (document containing XML tags and content) with correct syntax as defined by the XML standard is referred to as being "well-formed."
- "Well-formed" refers only to the ***structure*** and ***syntax*** of the XML.

Well-formed XML *(2 of 2)*

- A "well-formed" XML document instance has correct XML syntax.
- It is a document that conforms to the XML syntax rules.
- A well-formed document is one that meets the minimum criteria for XML parsers to read the document.

Rules for Well-formed XML *(1 of 2)*

- XML declaration (Prolog) must come first in every document.
- Comments are not valid within a tag.
- Comments may not contain two hyphens in a row other than the beginning and end of the comment.
- Tags must have an end tag, or be closed within the singleton tag itself, for example **<PersonAge/>**

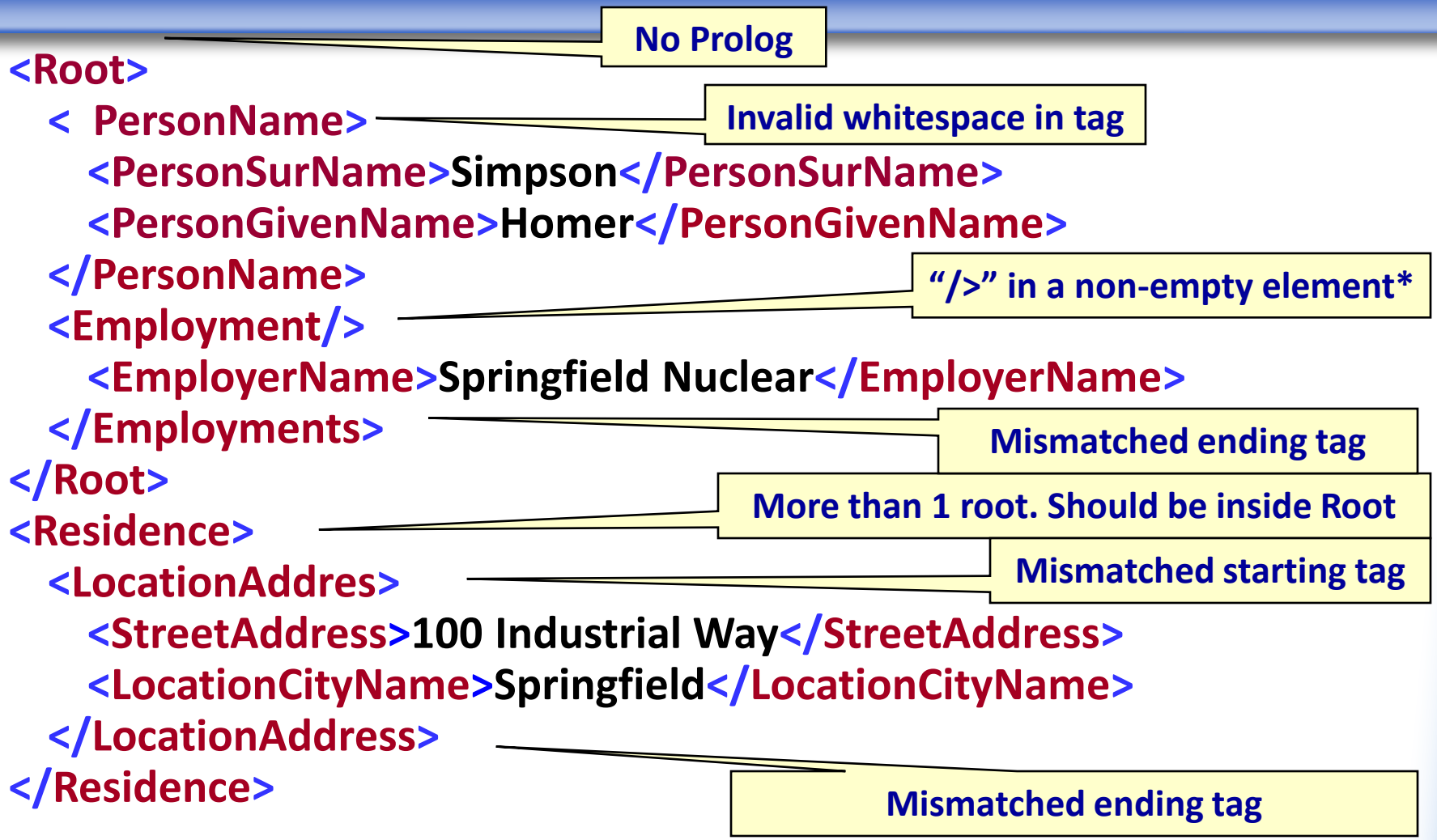
Rules for Well-formed XML *(2 of 2)*

- All attributes of tags must be quoted.
  - Preferably double quotes, unless attribute contains a double quote (then use single quotes).
- Every XML document must contain one root element that completely contains all the other elements.
- Tags cannot overlap – must nest correctly.

Exercise 2.2: Find the Errors

```
<Root>
  <  PersonName>
    <PersonSurName>Simpson</PersonSurName>
    <PersonGivenName>Homer</PersonGivenName>
  </PersonName>
  <Employment/>
    <EmployerName>Springfield Nuclear</EmployerName>
  </Employments>
</Root>
<Residence>
  <LocationAddres>
    <StreetAddress>100 Industrial Way</StreetAddress>
    <LocationCityName>Springfield</LocationCityName>
  </LocationAddress>
</Residence>
```

Solution 2.2: Find the Errors



What is an XML Instance?

- When you use the rules of an XML schema to create content, the result is an XML document instance.
- The schema controls what the instance looks like and what is in it.
  - i.e., it is the blueprint for what an XML instance can look like.

PersonName Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetnamespace="http://myNamespace.com"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="PersonNameType" >
      <xs:sequence>
        <xs:element name="PersonNamePrefix" type="xs:string" minOccurs="0"/>
        <xs:element name="PersonGivenName" type="xs:string" minOccurs="0"/>
        <xs:element name="PersonMiddleName" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="PersonSurName" type="xs:string"/>
        <xs:element name="PersonSuffixName" type="xs:string" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="PersonNameCode" use="optional" default="RealName">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="RealName"/>
            <xs:enumeration value="Alias"/>
            <xs:enumeration value="BirthName"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
    <xs:element name="PersonName" type="PersonNameType"/>
</xs:schema>
```

Resulting Example Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<PersonName PersonNameCode="RealName">
   <PersonNamePrefix>Mr</PersonNamePrefix>
   <PersonGivenName>Juan</PersonGivenName>
   <PersonMiddleName>Anna</PersonMiddleName>
   <PersonSurName>Tu</PersonSurName>
   <PersonSuffixName>Sr</PersonSuffixName>
</PersonName>
```

Exercise 2.3: Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:local-ns="http://myNamespace.com"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Location">
      <xs:complexType>
        <xs:sequence>
            <xs:element name="LocationName" type="xs:string" minOccurs="0"/>
          <xs:element name="LocationAddress">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="AddressFullText" type="xs:string" minOccurs="0"/>
                <xs:element name="LocationCityName" type="xs:string" minOccurs="0"/>
                <xs:element name="LocationStateName" type="xs:string"/>
                <xs:element name="PostalCodeText" type="xs:string" minOccurs="0"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

Solution 2.3: Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<Location
   xmlns:my="http://myNamespace.com"
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xsi:schemaLocation="Path to schema" >

   <LocationName>Charville Farmhouse</LocationName>
   <LocationAddress>
      <AddressFullText>421 Bath Rd</AddressFullText>
      <LocationCityName>Charville</LocationCityName>
      <LocationStateName>Utah</LocationStateName>
      <PostalCodeText>83405</PostalCodeText>
   </LocationAddress>
</Location>
```

Module Summary

- In this module, we reviewed how to:
  - Compare and contrast XML elements and attributes.
  - Describe the characteristics of a well-formed instance.
  - Create an XML instance based on the given XML schema.

Creative Commons

creative commons
COMMONS DEED

Attribution-ShareAlike 2.0

You are free to

- Copy, distribute, display, and perform the work
- Make derivative works
- Make commercial use of the work

Attribution—You must give the original author credit

ShareAlike—If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one

Under the following conditions

- For any reuse or distribution, you must make clear to others the license terms of this work
- Any of these conditions can be waived, if you get permission from the copyright holder

Your fair use and other rights are in no way affected by the above

This is a human-readable summary of the Legal Code (the full license) and Disclaimer

This page is available in the following languages
Català, Deutsch, English, Castellano, Suomeksi, français, hrvatski, Italiano, 日本語, Nederlands, Português, and 中文(繁)

Learn how to distribute your work using this license